Project: Build a Traffic Sign Recognition Program

Overview

In this project, you will use what you've learned about deep neural networks and convolutional neural networks to classify traffic signs. You will train and validate a model so it can classify traffic sign images using the German Traffic Sign Dataset (http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset).

After the model is trained, you will then try out your model on images of German traffic signs that you find on the web.

The Project

The goals / steps of this project are the following:

- Load the data set
- Explore, summarize and visualize the data set
- Design, train and test a model architecture
- Use the model to make predictions on new images
- Analyse the softmax probabilities of the new images
- Summarize the results with a written report

Data Set Summary & Exploration

I used the pandas library to calculate summary statistics of the traffic signs data set:

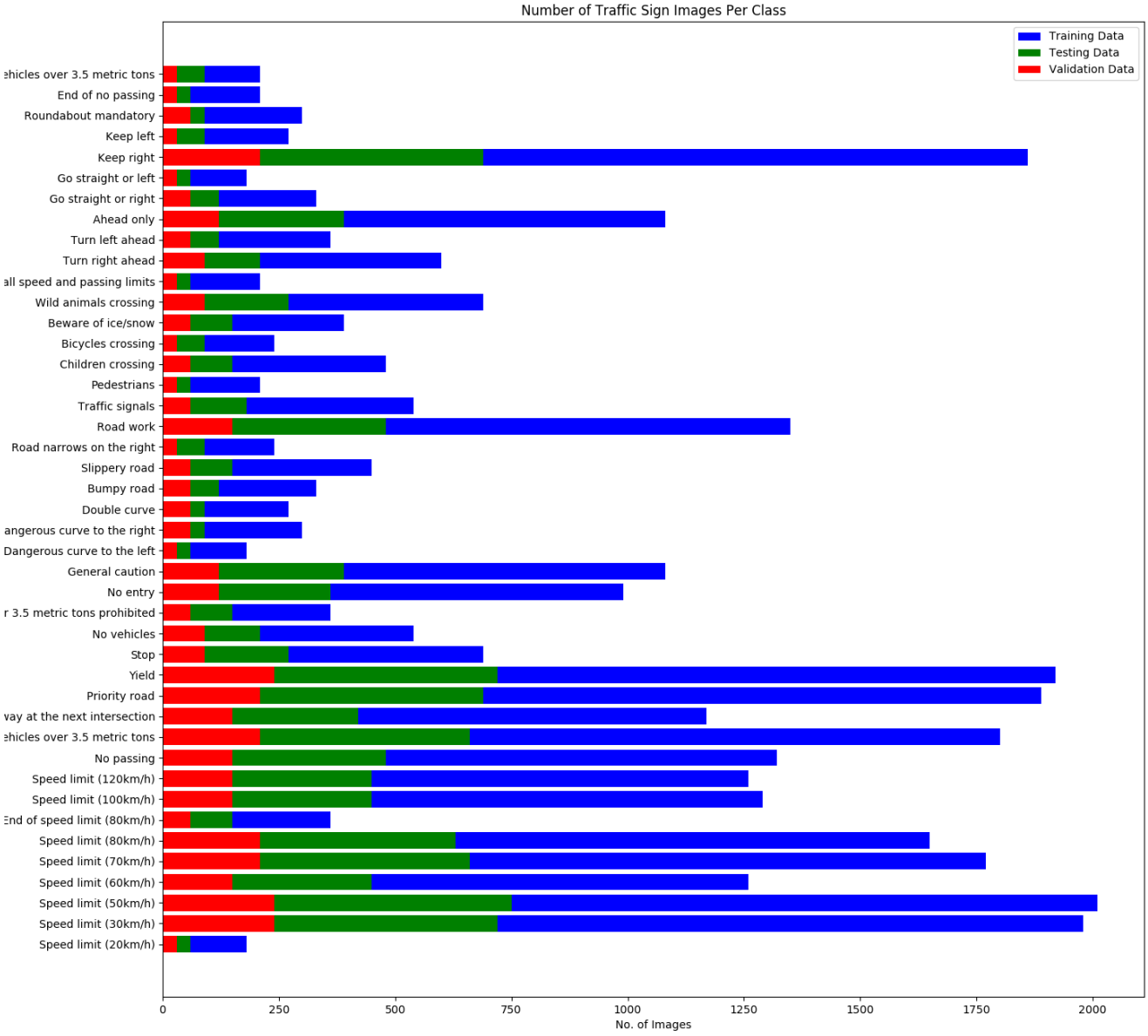
- The size of training set is 34799.
- The size of the validation set is 34799.
- The size of test set is 12630.
- The shape of a traffic sign image is 32x32x3.
- The number of unique classes/labels in the data set is 43.

Exploratory visualization of the dataset

Below is an exploratory visualisation of the data set.

It is a bar chart showing how many images there are for each of the 43 labels.

It is clear that the data is not evenly balanced across the labels, the range of data samples per label being between 180 to 2320. This can lead to poor training of some labels in comparison to others.

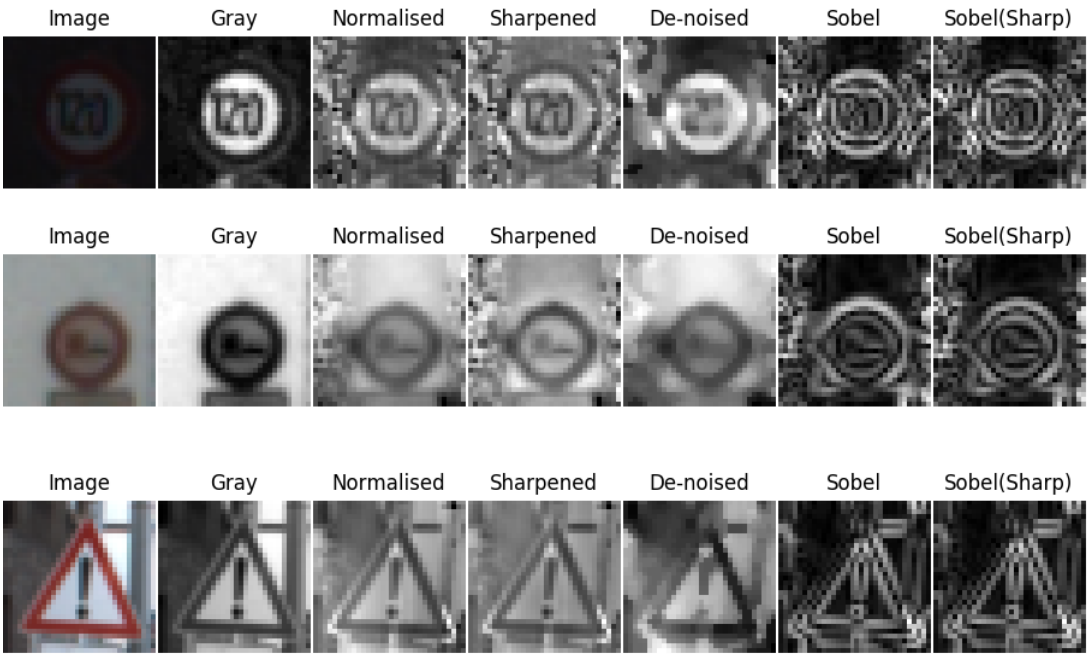


Design and Test a Model Architecture

Pre-processing

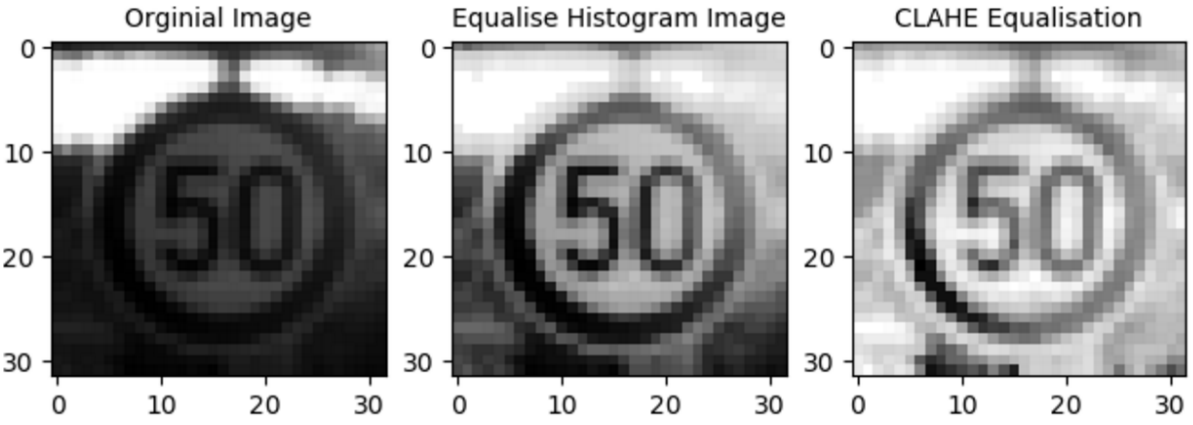
As a first step, I decided to convert the images to grayscale to reduce the number of layers that would be processed. I tried running the model on both RGB and Grayscale images and found no improvement with RGB, in fact generally the model was not as suited to the RGB as the grayscale. Furthermore, the grayscale images were faster to train with.

Here are some examples of a traffic sign images undergoing some of the various pre-processing cases including gray scale, normalisation, sharpening, de-noise and Sobel filters.



I tried several different techniques for the normalisation and initially found the pre-processing scale function most consistent / reliable. Other normalisation methods shows significant white or black spots within the images at times. I have since add contrast limited adaptive histogram equalisation and image sharpening which has increased the validation accuracy.

Sharpening the image at times presented visually better images, and other times less visually discernible. In general, sharpening the images added a percentage improvement in accuracy.



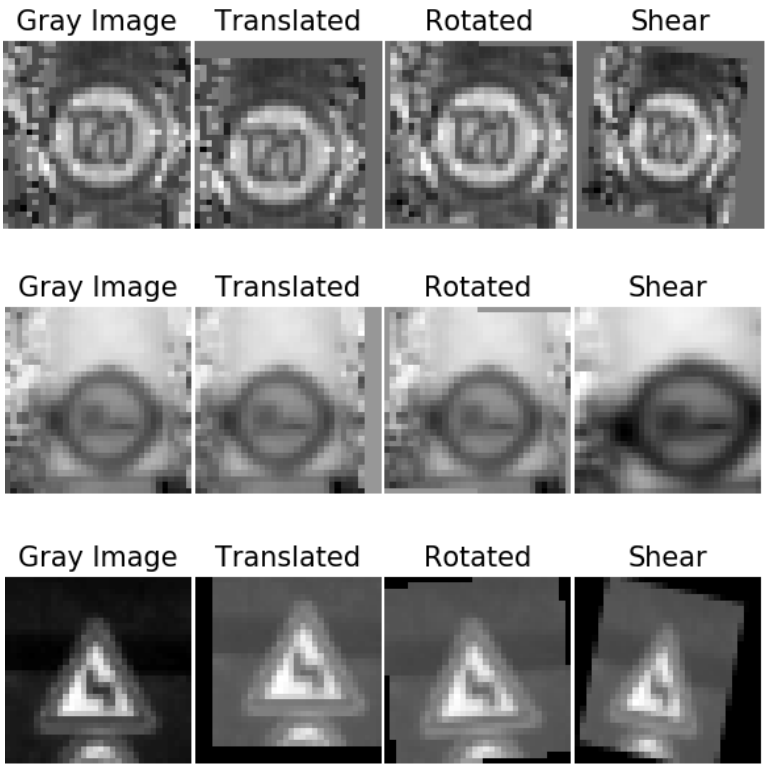
Data Augmentation

I decided to generate additional data because I was worried the model would bias training for those with more data. When checking validation accuracy, it will be worth noting which images have lower accuracy and what correlation there is to the number of images originally and the effect on data augmentation.

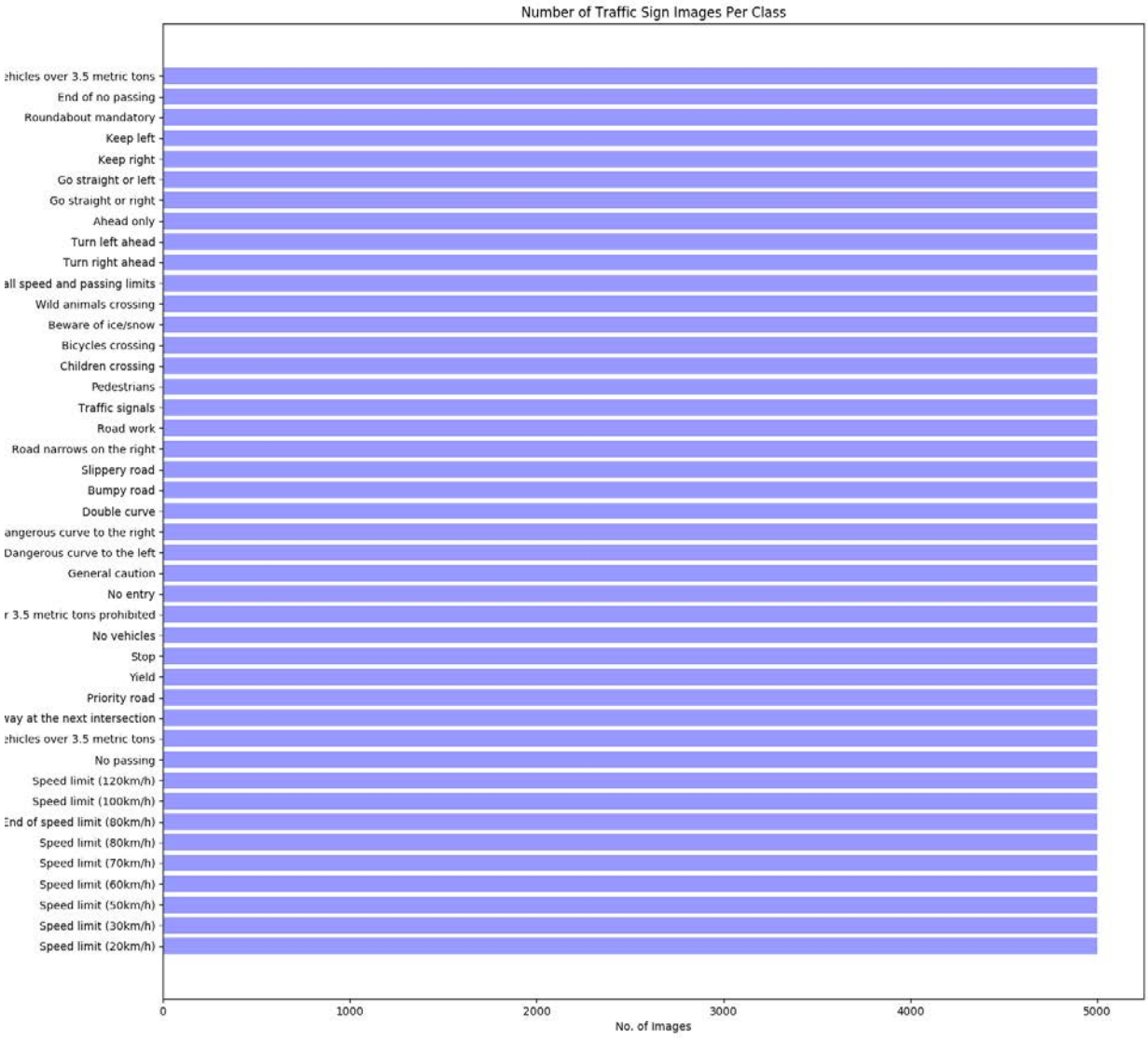
To add more data to the data set, I performed a series of transformations including rotations, translations and shear. The amount of rotation, translation or shear was determined randomly, however the amount of transformation was limited so that the images were not confused. It is unlikely to find signs upside down, back to front and only half in image. To prevent stack multiple

transformations on top of another only the original data set is used for augmentation. The number of images per bin can be specified. I tried adding noise and blurring images, however I found many images were already very blurred and noisy and I did not want to further affect the image quality.

Here is an example of an original image and an augmented image:



For a given model I tried increasing the number of augmented data images for each bin to 2000, 2500, 5000 and 10000. I found an improvement from 2000 to 2500, however not further improvements in accuracy to 5000 or 10000.



Model architecture

Several model architectures were created.

After doing some research I learnt about the Inception V4 used in GoogleNet for image classification. The GoogleNet is very complex and I only implemented two layers to test my understanding on the inception construction. It took a lot of time to tune parameters before the model would train most notably the learning rate and dropout. This model did not work well until augmented data was added.

I decided this model was too complex and time consuming so I began modifying the LeNet model. It was easy to modify for the 32x32 gray scale images and run with descent accuracy. After adding another convolution layer and some depth I was able to reach 97.3% accuracy on the validation set. The training process was significantly faster. I had difficult improving the accuracy further so I decided to try a compromise between the 2 layer inception model and the LeNet. I used a single layer inception with much greater depth. The final validation accuracy is 99.0% and the test accuracy was 97.3%.

It was difficult to understand the best way to modify the architecture to make improvements for the traffic signs. Training the models was largely trial and error. I systematically modified one parameter at a time and observed changes in accuracy and fit. Plotting the training and validation accuracy was important in understanding whether the model was underfitting or over fitting and how fast it was performing.

I found large batch sizes would help to smooth the learning curve and speed the training, however at times it would also reduce the accuracy and cause over fitting. I have generally used a batch size of 128.

I typically set Epoch values to 100, if I noticed it had converged I would stop training.

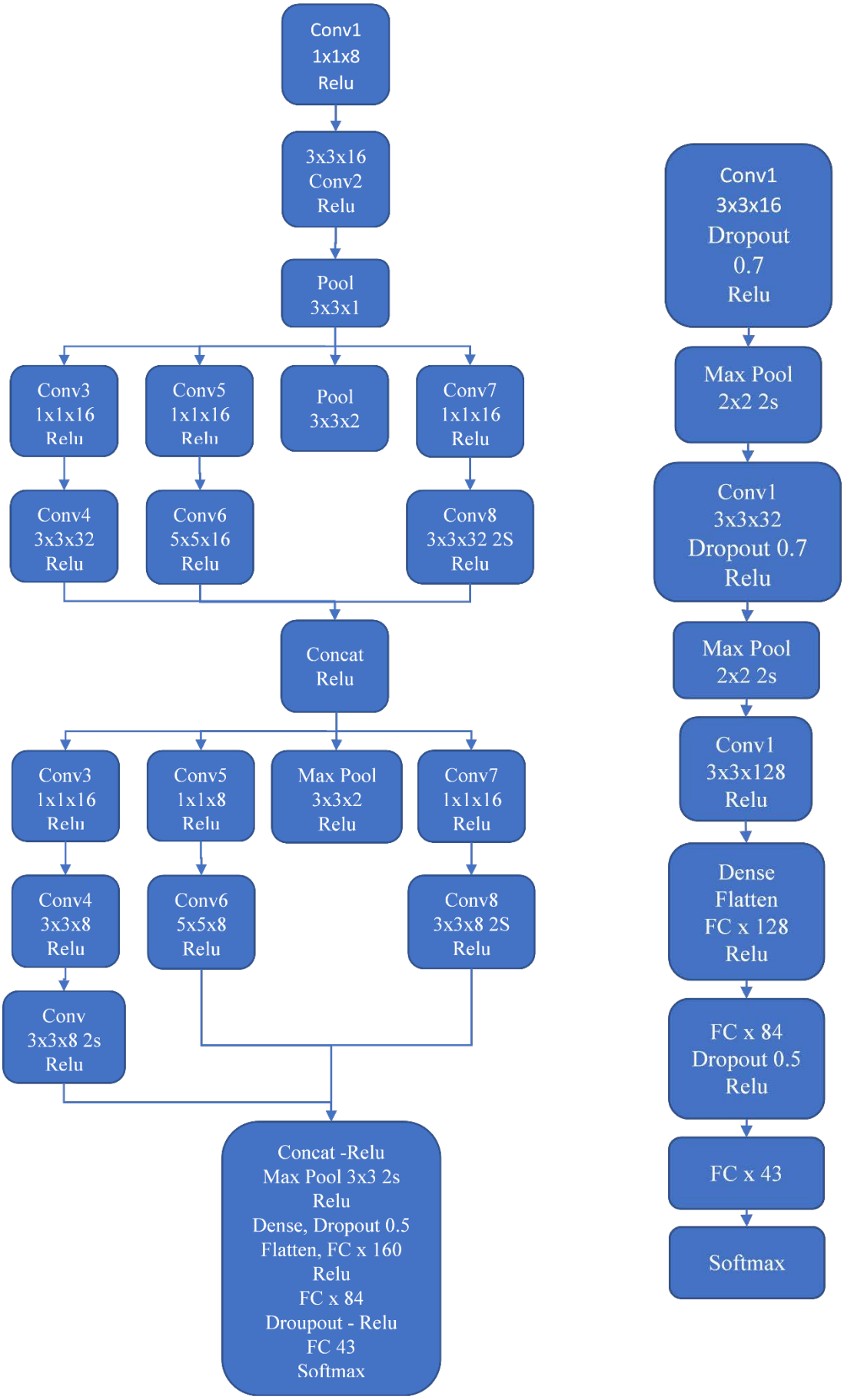
One of the more challenging aspects with training the models has been overfitting. Following research into how to prevent over fitting, a common approach is to apply drop out after each layer. It was learnt that using a lower drop out rate for the early convolution layers and a high drop out rate for the later layers can prevent the model from failing. It was also found that making the depth deeper and adding a convolution layer increased the accuracy most. Unfortunately adding the extra layer and depth also increase overfitting.

I experimented with various dropouts, max pool, average pool and relu. I found most models were over fitting so they were applied frequently. The dropout was more beneficial for the more complex inception models, whilst in the LeNet model it would reduce accuracy.

I choose the Adam optimiser based on discussions and recommendations in the community and from past exercise. The Adam optimiser has an adaptive learning rate for useful for fast convergence and training on deep or complex neural networks.

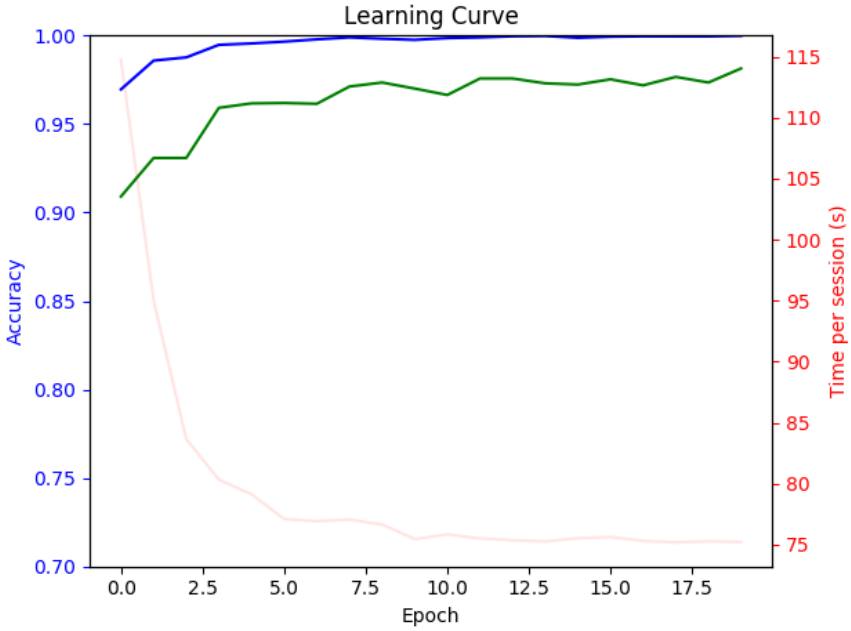
The two layer inception and modified LeNet (LeTrafNet) models can be seen on the following page.

Following each change in architecture, changes to the learning rate were often performed, particularly if the learning curve was jumpy in nature.

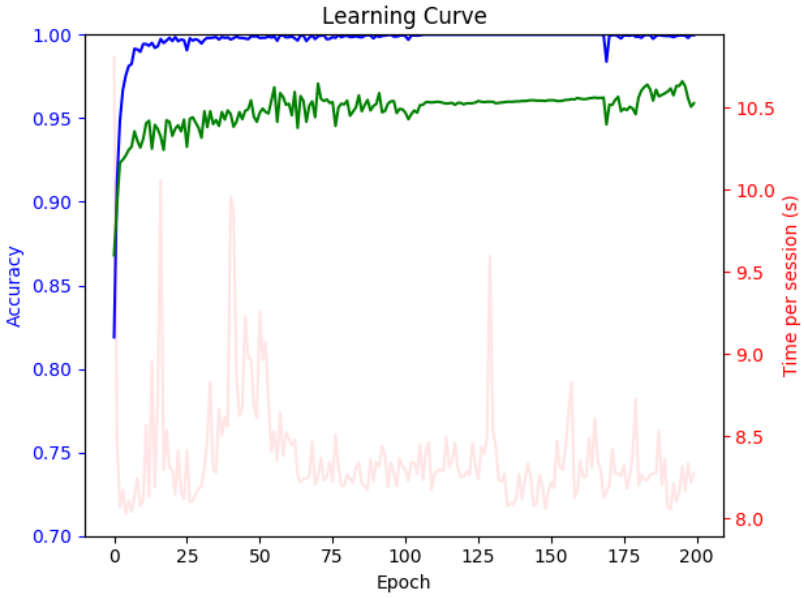

Conv1
1x1x8
Relu
3x3x16
Conv2
Relu
Pool
3x3x1
Conv3
1x1x16
Relu
Conv5
1x1x16
Relu
Pool
3x3x2
Conv7
1x1x16
Relu
Conv4
3x3x32
Relu
Conv6
5x5x16
Relu
Conv8
3x3x32 2S
Relu
Concat
Relu
Conv3
1x1x16
Relu
Conv5
1x1x8
Relu
Max Pool
3x3x2
Relu
Conv7
1x1x16
Relu
Conv4
3x3x8
Relu
Conv6
5x5x8
Relu
Conv8
3x3x8 2S
Relu
Conv
3x3x8 2s
Relu
Concat -Relu
Max Pool 3x3 2s
Relu
Dense, Dropout 0.5
Flatten, FC x 160
Relu
FC x 84
Droupout - Relu
FC 43
Softmax
Conv1
3x3x16
Dropout
0.7
Relu
Max Pool
2x2 2s
Conv1
3x3x32
Dropout 0.7
Relu
Max Pool
2x2 2s
Conv1
3x3x128
Relu
Dense
Flatten
FC x 128
Relu
FC x 84
Dropout 0.5
Relu
FC x 43
Softmax

Results

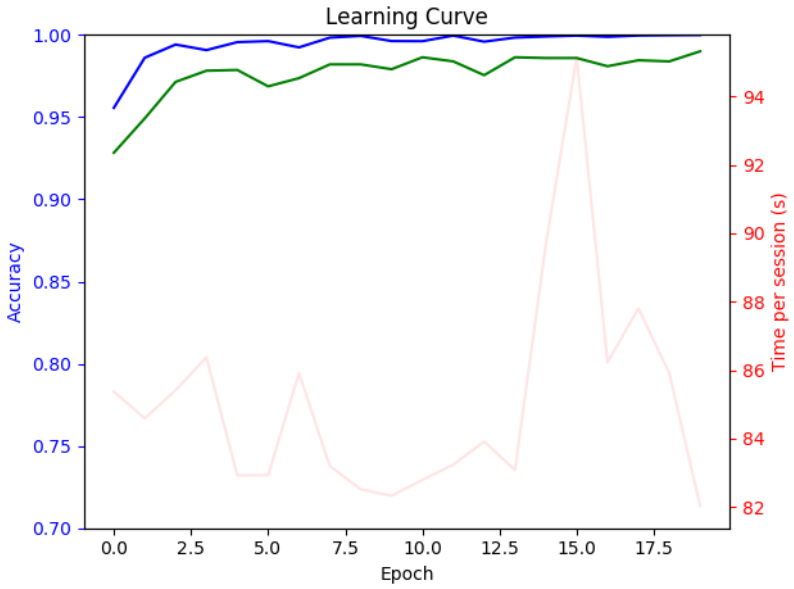
```
1 Layer Inception Model without
- EPOCH 20 ...
Sess Time: 118.2sec
Validation Accuracy = 98.1%
Training Accuracy = 100%
Test Accuracy = 96.6%
```




```
EPOCH 200 ... LeTrafNet
Sess Time: 8.3sec
Validation Accuracy = 95.9%
Training Accuracy = 100%
Test Accuracy = 94.7%
```



```
1 Layer Inception Model with CLAHE and Sharpening
- EPOCH 20 ...
Sess Time: 82.0sec
Validation Accuracy = 99.0%
Training Accuracy = 100%
Test Accuracy = 97.3%
```

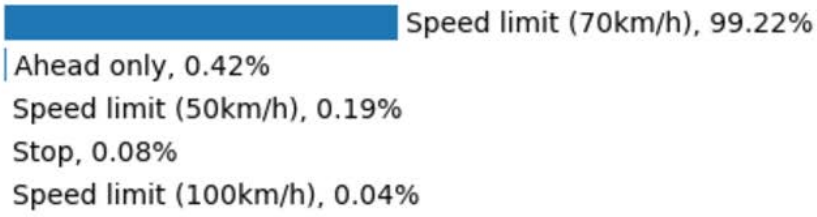


Online Test Images

A subset of the following images were used and passed through the saved model. The images were passed through the same pre-processing pipeline being grayscaled and normalised.

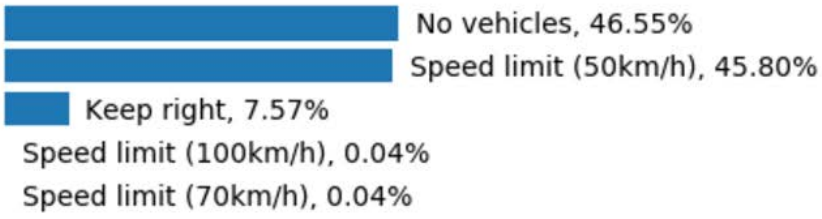
The session was run us tf.nn.top_k to determine the top 5 probabilities for each image of the models predictions. The results below are a selection from the trained single layer Inception model.

Not Trained

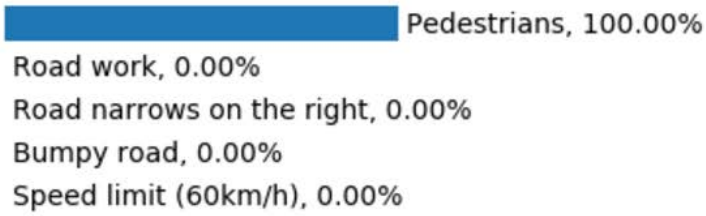


Incorrect: No vehicles

Not Trained



Correct: Pedestrians



Not Trained

Beware of ice/snow, 99.97%
Bumpy road, 0.02%
Bicycles crossing, 0.00%
Speed limit (60km/h), 0.00%
Priority road, 0.00%

Incorrect: Priority road

Speed limit (50km/h), 72.20%
Speed limit (70km/h), 15.70%
Speed limit (80km/h), 4.78%
Speed limit (100km/h), 4.32%
Speed limit (60km/h), 1.43%

Correct: Yield

Yield, 100.00%
No vehicles, 0.00%
Speed limit (20km/h), 0.00%
Speed limit (30km/h), 0.00%
Speed limit (50km/h), 0.00%

Not Trained

No entry, 100.00%
Stop, 0.00%
Turn left ahead, 0.00%
Speed limit (100km/h), 0.00%
Priority road, 0.00%

Correct: Turn left ahead

Turn left ahead, 97.65%
Stop, 2.20%
Keep right, 0.14%
Speed limit (120km/h), 0.01%
Ahead only, 0.00%

The model could correctly predict 10 of the 16 classified traffic signs, which gives an accuracy of 62.5%, in comparison the 96.6% with the test data, thus it seems the model is overfitting. I expected this to be greater due to the quality of the images provided. Perhaps because only real images were trained, these icons are not identifiable. I believe having at least blue and red colour channels available would be beneficial to recognising some images as these colours make a noticeable difference in a sign being a "do something" (blue) or "don't do something" (red) sign. Some of the incorrectly identify signs have somewhat resemblance and some of the second highest probabilities are correct. However, it should be noted that the first probabilities are very high in comparison to others, I suspect this is due to the dropouts, relu and softmax reducing the probabilities of other predictions. Despite improvements in the network using CLAHE and sharpening there was little difference in the prediction accuracy for new images, in fact the same images were predicted wrongly.

Summary

- 6 incorrect predictions
- 10 correct predictions
- 10 predictions not learnt
- 62.5% % accuracy

It is interesting to note which images have been predicted incorectly. The Priority Road has been mistaken as a speed limit sign, resembling the same shape. Perhaps if the red color was used for training it would help in classifying the sign.

The Narrow Roads sign has been mistaken for Traffic Signals, this is likely due to the little data provided for this image class. There perhaps better data augmention and data collection is required.

The slippery road sign has been mistaken for beware of ice/snow. Both of these data sets did not have significant images available either, they also have similar resemblence in shape. The prediction does not fall into the top five either.

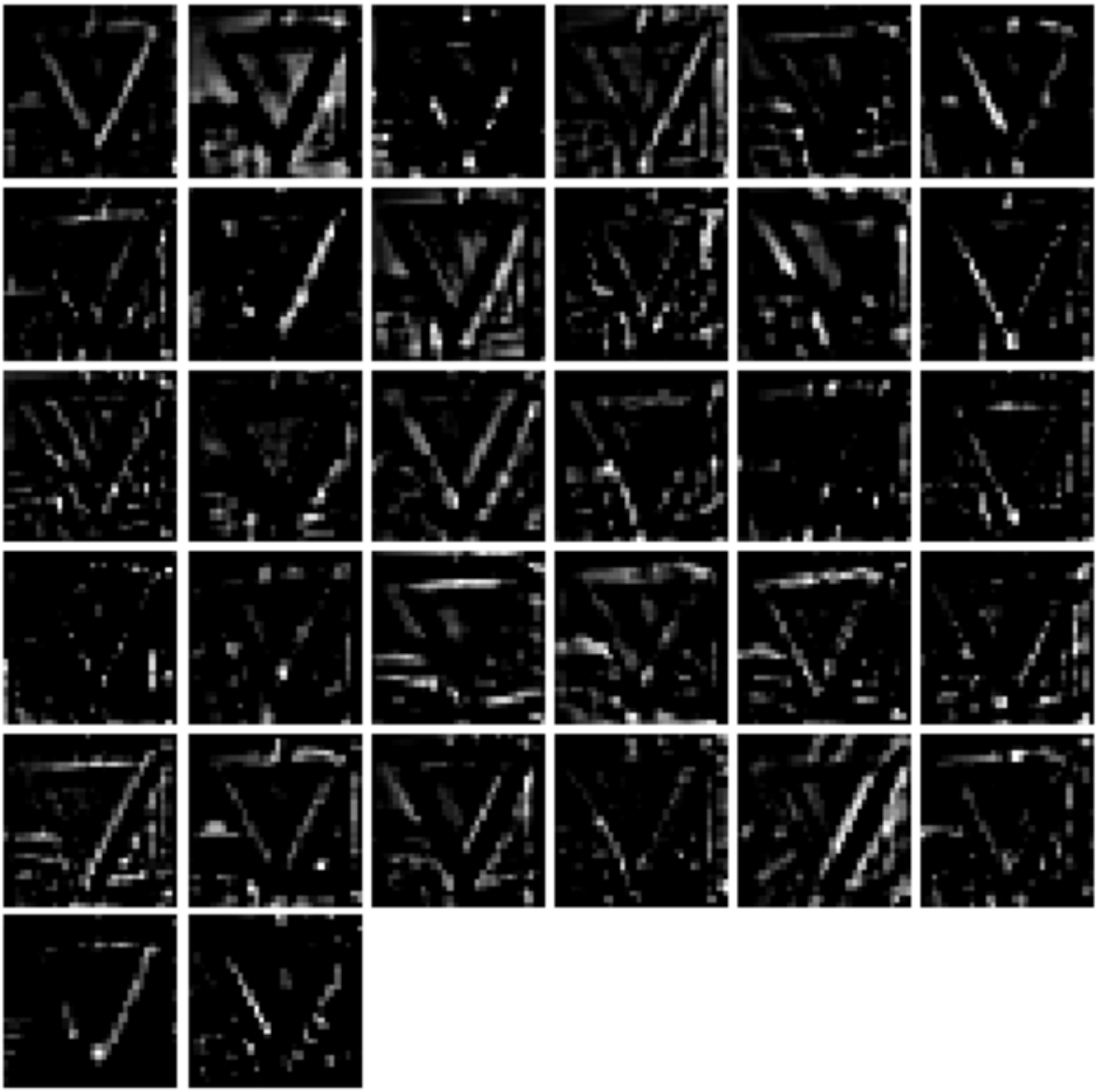
The yield to round about sign has been misclassified, it should be noted the amount of data available for this sign is also at the lower end. For this sign I believe colour would also be a useful classifier.

Visualise the Neural Networks State with Test Images

In order to better understand the output of the neural networks weights the feature maps for each activation layers have been plotted.

The feature map for the first layer appears to highlight edges in the image. The feature map for the second layer can be seen below, it appears to highlight individual lines on sign.

The feature map for the third layer appears to highlight vertices. It is not clear to me what other feature maps



Conclusion

This report has looked at

- Loading data sets
- Exploring, summarising and visualising data sets
- Designing, training and testing model architectures
- Using the model to make predictions on new images
- Analysing the softmax probabilities of the new images
- Summarising the results with a written report
- Visualising the feature maps for the convolutional filters

Future Work

Future work on this topic will involve explore masking the rgb images such that only white, blue and red colours are used in training the Neural Network.